Listing of Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method for executing a target application on a computer, comprising the steps of:

executing program code of said target application;

receiving a notification of an event specifying an unexpected or erroneous behavior in execution of said program code;

searching a configuration file that is external to said target application, said configuration file maintains a listing of events and associated actions, at least one of said actions being a restarting of said target application when said target application becomes idle;

said configuration file specifies periodic checking for a thread starvation condition;

retrieving an action from said configuration file that is associated with said event; and

carrying out said action.

2. (Original) The method of claim 1 where said configuration file includes a plurality of events and associated actions, where said actions are taken from a set that includes thread restarts.

3. (Original) The method of claim 1 where said configuration file includes a plurality of events and associated actions, where said actions are taken from a set that includes a partial state dump.

4. (Original) The method of claim 1 where said configuration file includes a plurality of events and associated actions, where said actions are taken from a set that restarting said target application from a checkpointed state.

5. (Original) The method of claim 1 where said configuration file includes a plurality of events and associated actions, where said actions are taken from a set that restarting a thread of said target application from a checkpointed state.

6. (Canceled)

7. (Previously Presented) The method of claim 1 further comprising a step of periodically checking for the thread starvation condition.

8. (Original) The method of claim of claim 7 where said checking for a thread starvation condition includes the step of checking whether there is a subset of threads of said target application that take more than a predetermined share of CPU time of said computer.

9. (Original) The method of claim 7 where said checking for a thread starvation condition includes the step of checking whether a thread of said target application receives less than a predetermined share of CPU time of said computer.

10. (Previously Presented) The method of claim 7 where said checking for a thread starvation condition is carried out with a thread that

records a time t1 for relinquishing CPU time of said computer;

relinquishes CPU time of said computer for a specified time,

records a time t2 when it reacquires CPU time of said computer, and

concludes that the thread starvation condition exists when time t2 is greater than time t1 by a predetermined amount.

11. (Previously Presented) The method of claim 8 where said configuration file includes a plurality of events and associated actions, where said actions are taken from a set that includes recovery from the thread starvation condition.

12. (Previously Presented) The method of claim 11 where said recovery from the thread starvation condition comprises suspending one or more threads for a preselected period of time.

13. (Original) The method of claim 12 where said suspending one or more threads is carrying out by iteratively selecting a thread, suspending the selected thread, and testing effect of the suspension of the selected thread on said thread starvation condition.

14. (Original) The method of claim 12 where said one or more threads that are suspended are threads that eliminate said thread starvation condition, or reduce the thread starvation condition by a predetermined amount.

15. (Original) The method of claim 1 further comprising a step checkpointing said target application in accordance with a predetermined algorithm, and said configuration file includes at least one action that restarts said target application based on information obtained via said checkpointing.

16. (Original) The method of claim 1 further comprising a step checkpointing said target application that is executed at regular intervals, or when the amount of information stored for said target application exceeds a predetermined threshold, or when activity level of said target application exceeds a predetermined threshold.

17. (Original) The method of claim 1 where said configuration file includes a plurality of events and associated actions, where said actions are taken from a set that includes a checkpointing and a restart based on information obtained from said checkpointing.

18. (Original) The method of claim 17 where said checkpointing is in accordance with information provided by said program code.

19. (Original) The method of claim 18 where said program code specifies a thread and all progeny of said thread.

20. (Original) The method of claim 1 further comprising a step checkpointing when said notification of said event is received, and said configuration file includes at least one action that restarts said target application based on information obtained via said checkpointing.

21. (Original) The method of claim 20 where said checkpointing stores state information in accordance with specification by said target application.

22. (Currently Amended) A system including a processing unit including a target application executing on said processing unit; and a supervisor software module external to said target application and executing on said processing unit characterized in that:

execution code of said target application is unaware of said supervisor module; and

said supervisor module monitors execution of said target application, and in response to an error or unexpected behavior in said execution, takes action that affects said execution, which action is taken from a set of actions that includes an action for terminating execution of said software module and at least an action that restarts said target application only when said target application becomes idle, and determines whether said error exists by checking implicit conditions including at least a thread starvation condition in said execution of said software module.

23. (Original) The system of claim 22 where said set includes restarting an execution thread of said application.

24. (Canceled)

25. (Canceled)

26. (Original) The system of claim 22 where said set includes storing in a file a partial state of said target application.

27. (Original) The system of claim 22 where said set includes storing in a file state information of a thread of said target application.

28. (Original) The system of claim 22 where said set includes checkpointing.

29. (Original) The system of claim 28 where said checkpointing is triggered by code included in said target application.

30. (Original) The system of claim 23 where said set further includes taking no action relative to execution of said software module, but incrementing an error counter and taking one or more of the following actions:

terminating said application when said error counter reaches a preselected threshold;

restarting said application immediately when said error counter reaches a preselected threshold;

restarting said application when it becomes idle when said error counter reaches a preselected threshold;

terminating an execution thread of said application when said error counter reaches a preselected threshold; and

restarting an execution thread of said application when said error counter reaches a preselected threshold.

31. (Original) The system of claim 22 where said action taken by said supervisor is retrieved from a configuration file that is specific to said target applications, which file specifies actions to be taken in response to specified signal reporting on said error or unexpected behavior.

32. (Original) The system of claim 31 where said configuration file is modifiable by a user of said application module.

33. (Original) The system of claim 31, further comprising a configuration manager that creates said configuration file by evaluating said code of said application module.

34. (Original) The system of claim 22 where said supervisor interacts with a checkpointing module before taking an action from said set that restarts execution of said application module.

35. (Original) The system of claim 34 where said checkpointing module that checkpoints execution is independent of said supervisor.

36. (Original) The system of claim 35 where said independent module checkpoints pursuant to a predetermined algorithm of said checkpointing module.

37. (Original) The system of claim 34 further comprising a configuration file that specifies errors that restart execution of said application with information developed by said checkpointing.

38. (Original) The system of claim 34 further comprising a configuration file that specifies errors that restart execution of said application with information developed by said checkpointing and errors that restart execution of said target application without information developed by said checkpointing.

39. (Original) The system of claim 22 where said supervisor maintains state information of an execution thread of said target application, which when said execution thread causes said error or unexpected condition, said supervisor restarts said thread using said state information.

40. (Currently Amended) A system including a processing unit including an application software module executing on said processing unit; and a supervisor software module external to said application software module and executing on said processing unit characterized in that:

execution code of said application software ~~application~~ module makes no reference to said supervisor module except for sending one or more messages to said supervisor, at one or more locations of said code, specifying a subset of state

information of said application module for said supervisor module to keep track of for possible checkpointing; and

said supervisor module monitors execution of said application module, concurrently keeps tract of scope of said subset of state information specified in a most recently received one of said messages, and determines whether said errors exist by checking implicit conditions including at least a thread starvation condition during execution of said code.

41. (Original) The system of claim 40 where said supervisor stores checkpointing information, pursuant to said scope of said subset of state information when said supervisor determines that execution of said application module is characterized by abnormal behavior that calls for restarting execution of said application module.

42. (Original) The system of claim 41 where, following said storing of checkpointing information, said supervisor restarts execution of said application module.

43. (Original) The system of claim 40 where each of said one or more messages specifies said subset of state information by specifying one or more object trees.